Information Systems Concepts

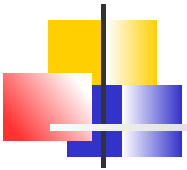
Fundamentals of Object Technology

Roman Kontchakov

Birkbeck, University of London

Based on Appendix of Maciaszek, L.A.:

Requirements Analysis and System Design (3rd Edition) Addison Wesley, 2007.

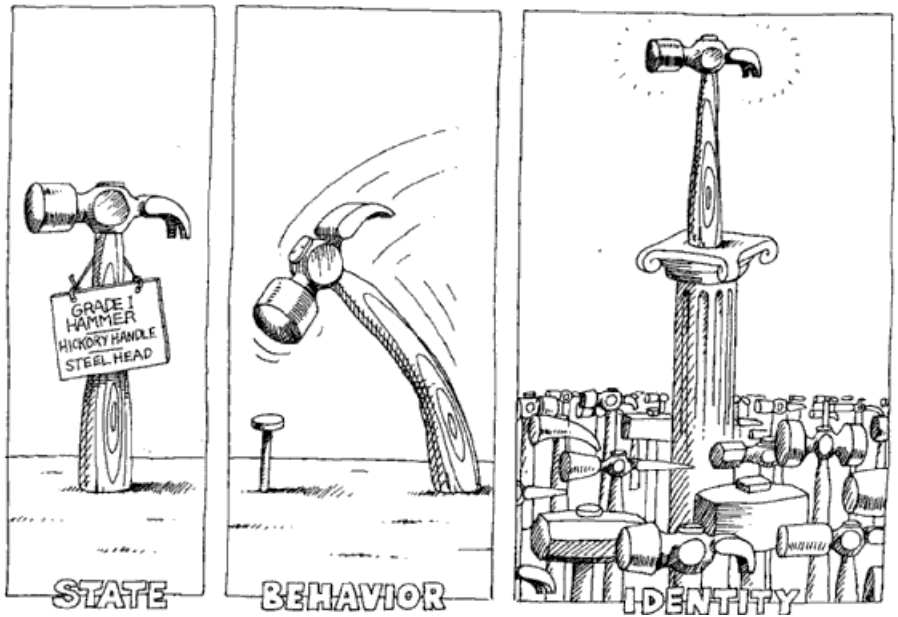


Outline

- Object
- Class
- Association
- Aggregation and Composition
- Generalization
- Inheritance
- Polymorphism

Object: State, Behaviour & Identity

NB: equal
≠
identical

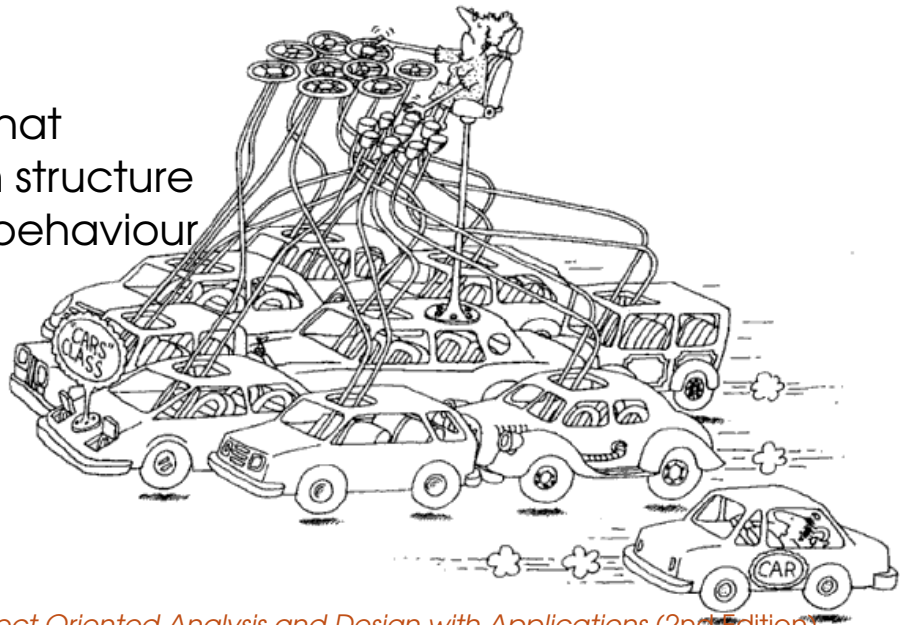


Booch, G.: *Object Oriented Analysis and Design with Applications* (2nd Edition) Addison-Wesley, 1994

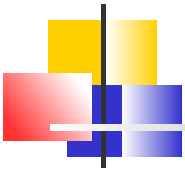
Class

Class represents
a set of objects that
share a common structure
and a common behaviour

Objects
are instances
of classes



Booch, G.: *Object Oriented Analysis and Design with Applications* (2nd Edition) Addison-Wesley, 1994



Objects: UML Notation

c1: Module
module_code = COIY016H4 module_name = Information Systems Concepts

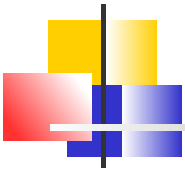
object-name: class-name

attribute-name = value

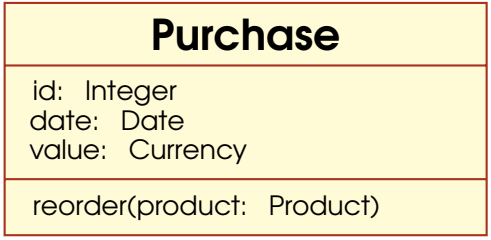
NB: '**:Module**' is an anonymous instance of class **Module**

NB: '**c1**' is an object without a specified class

NB: there is no compartment for operations!



Classes: UML Notation



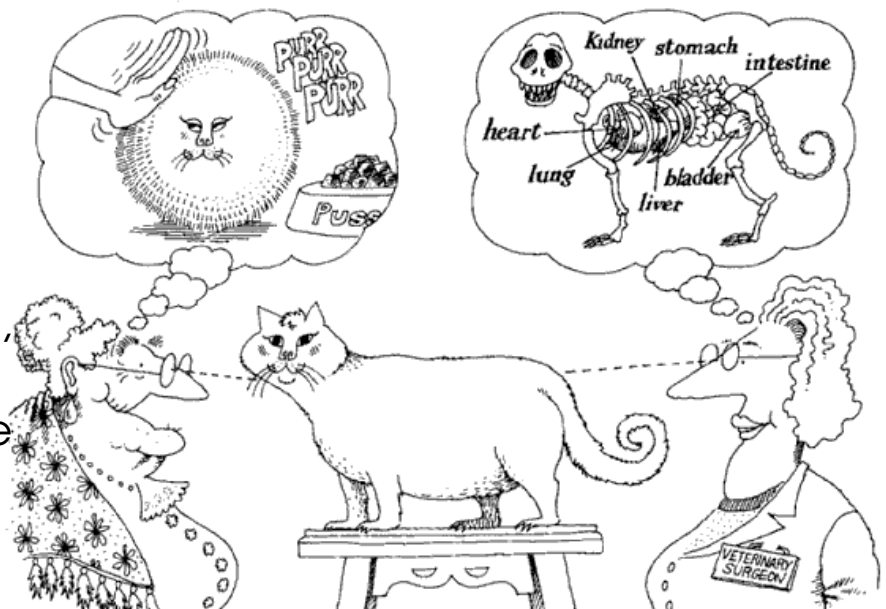
class name

attribute names and types

operation signatures

Abstraction

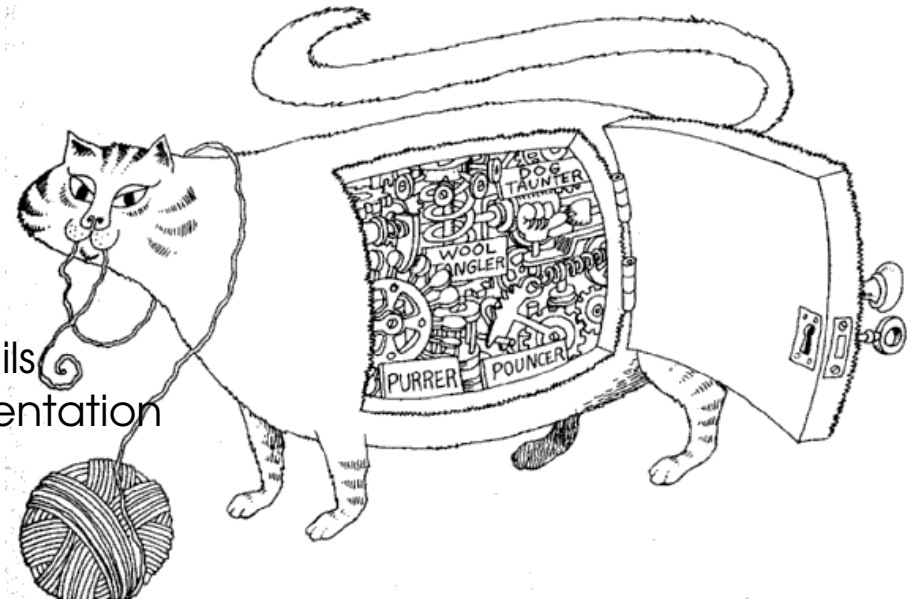
Abstraction focuses upon the essential characteristics of some object, **relative** to the perspective of the viewer



Booch, G.: *Object Oriented Analysis and Design with Applications* (2nd Edition) Addison-Wesley, 1994

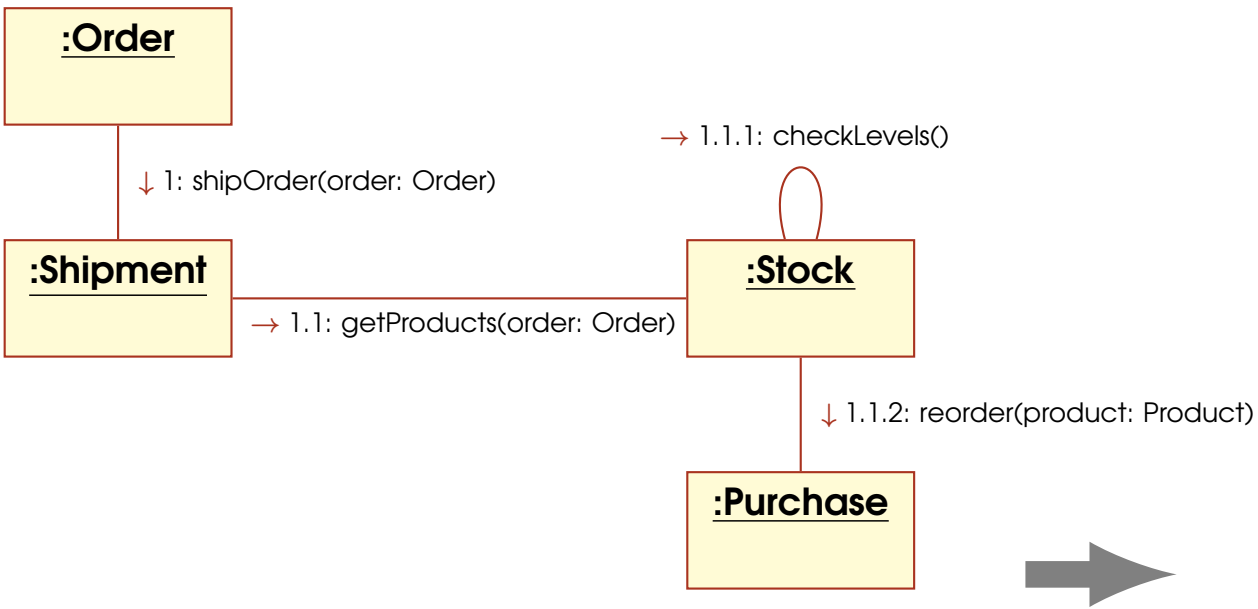
Encapsulation

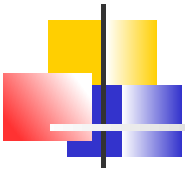
Encapsulation
hides the details
of the implementation
of an object



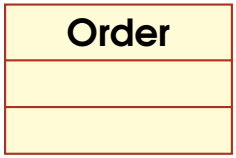
Booch, G.: *Object Oriented Analysis and Design with Applications* (2nd Edition) Addison-Wesley, 1994

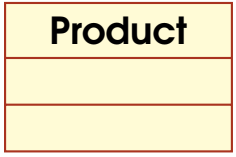
Operations (in Communication Diagrams)

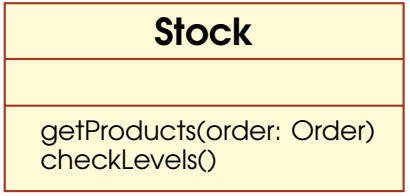


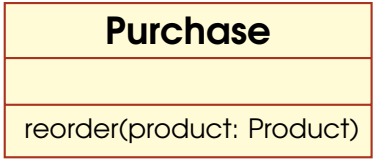


Operations (in Class Diagrams)







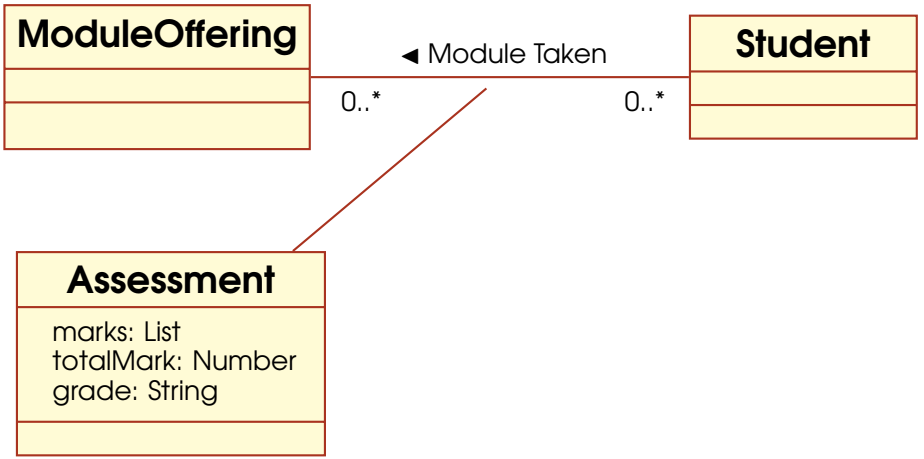


Associations

multiplicity:

- $\underbrace{0..1}$ none or one
- $\underbrace{0..*}$ any number
- $\underbrace{1..*}$ at least one
- $\underbrace{n}$ exactly n

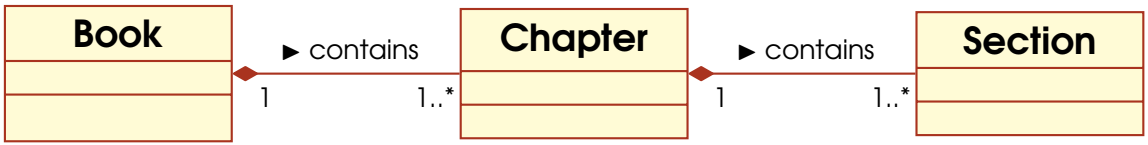
Association Classes



Composition & Aggregation

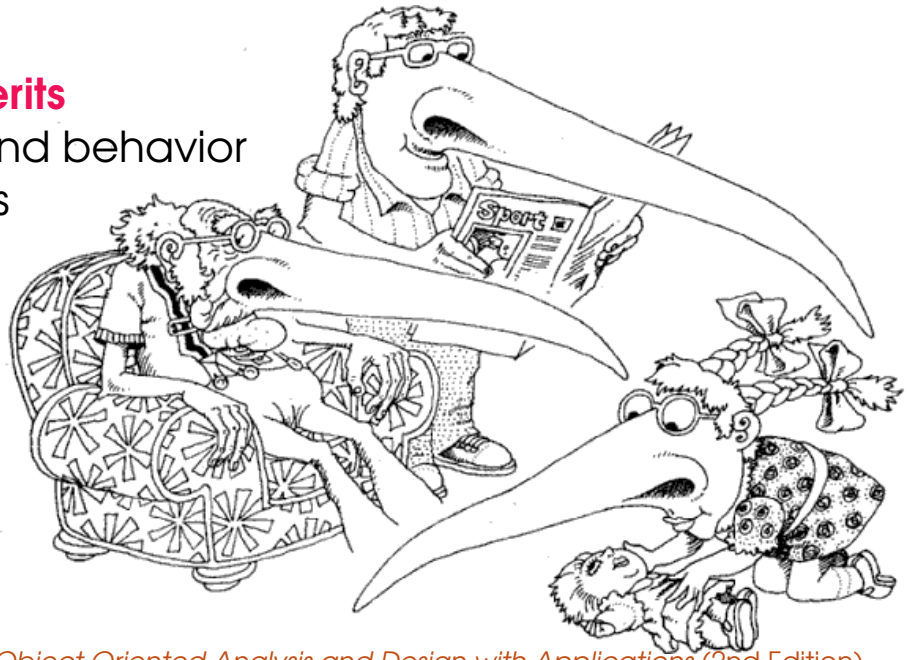
Aggregation: by reference (transitive and asymmetric)

Composition: by value (transitive, asymmetric and **existence-dependant**)



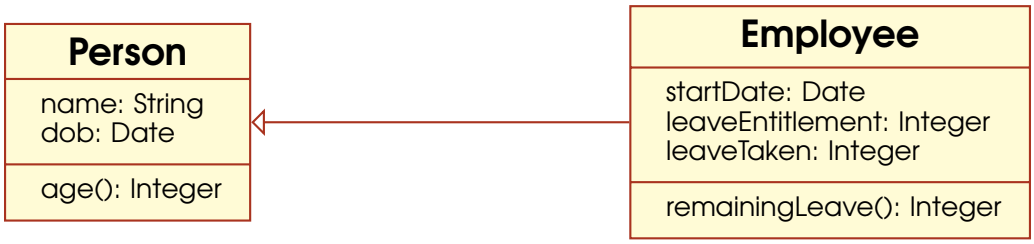
Generalization

A subclass **inherits**
the structure and behavior
of its superclass



Booch, G.: *Object Oriented Analysis and Design with Applications* (2nd Edition) Addison-Wesley, 1994

Inheritance in Java




```
public class Person
{
    private String name;
    private Date dob;

    public int age(){
        return getYear()
                - getYear(dob);
    }
}
```

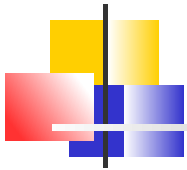
```
public class Employee extends Person
{
    private Date startDate;
    private int leaveEntitlement;
    private int leaveTaken;

    public int remainingLeave(){
        return leaveEntitlement - leaveTaken;
    }
}
```

Polymorphism in Java


```
public class Manager extends Employee
{
    private int leaveSupplement;

    public int remainingLeave(){
        int l = super.remainingLeave();
        return l + (leaveSupplement);
    }
}
```



Take Home Messages

- Each **object** has a state, behaviour and identity
- **Class** defines attributes and operations
- There are three kinds of relationships between classes:
  - association,
  - aggregation/composition and
  - generalization
- **Generalization** provides basis for inheritance and polymorphism